Hugo Documentation

Contents

Hugo

Hugo is a lightweight static site generator made with Golang. Hugo enables developers to create fast websites that can work with or without servers, this versatility of Hugo provides developers the flexibility to deploy Hugo websites on different environments or integrate them with already existing websites.

What are static websites?

Static websites are websites that do not rely explicitly on servers. Contents of static websites are not dynamically generated by servers but are rather a collection of predefined pages. The advantage of static websites over dynamic websites is the speed they offer to users and the ease of setup and deployment they offer to developers.

Given that Web Developers often begin their journey with HTML, CSS, and JavaScript with which static websites are built, they can start building websites without knowing serverside technologies.

Purpose of Hugo

Hugo allows users to manage static websites efficiently by providing them with useful commands that can be used to create, edit, and delete content with ease. It can be called a CMS for static websites. With Hugo, static websites can be managed with single or multiple commands without tempering with the existing design and configurations of websites.

In addition, Hugo has a repository of beautiful themes that can be customised to suit your needs. The community is vibrant, hence new themes both free and premium ones are published regularly.

Installation

This sections covers the installation steps to have Hugo installed on your machine. Hugo can be installed on your machine by following one of the below options:

- Download binaries.
- Build from source.
- Use package installers.

Download Binaries

Hugo binaries are cross-platform. Hence, you can download them and use them on your machines as you wish. Preferably, you should add the downloaded binaries to path to give you flexibility. Also, you can add the binaries on remote servers for cloud solutions.

Click here to download suitable binaries for your machines.

Build from Source

To build Hugo from source, you must have Golang installed. Download the latest version of Golang from https://go.dev if you do not have Golang installed on your machine. After you complete the basic Golang installation steps, follow the below steps to build Hugo from source on your machine:

MacOs users should run the following to build Hugo from source, first, you must install Golang on your Mac machine:

```
brew install go
```

After you install Golang, fetch the source code from GitHub and compile it.

```
    git clone https://github.com/gohugoio/hugo
```

```
    mkdir -p src/github.com/gohugoio
    ln -sf $(pwd) src/github.com/gohugoio/hugo
    go get
    go build -o hugo main.go
```

The generated hugo file is a binary that can added to path or moved about.

Use Package Installers

This method is recommended for less technical users and for technical users as well. Make sure you have the package installers on your machine before trying to install Hugo.

MacOS

For HomeBrew users, run the following commands to install Hugo:

```
brew install hugo
```

If you want only the absolute latest in development version, run the below:

```
brew install hugo --HEAD
```

You should see something similar to this if homebrew is working well:

```
     ==> Downloading https://homebrew.bintray.com/bottles/
hugo-0.21.sierra.bottle.tar.gz
     ######################################################################
 100.0%
     ==> Pouring hugo-0.21.sierra.bottle.tar.gz
     /usr/local/Cellar/hugo/0.21: 32 files, 17.4MB
```

Now to confirm if homebrew has successfully installed Hugo, run the following command:

```
hugo version
```

Windows

For Chocolatey users on windows, run the following command to install Hugo:

```
choco install hugo -confirm
```

Run the below command if you want Sass/SCSS support:

```
choco install hugo-extended -confirm
```

However, if you are using Scoop package manager, you should use the following command:

```
scoop install hugo
```

For an extended version, run:

```
scoop install hugo-extended
```

Linux

For Linux users, HomeBrew can be used as expressed for MacOS users. If you are using a Linux distribution that supports Snap, run the command below to install Hugo:

```
snap install hugo
```

For an extended version that supports Sass/SCSS, run:

```
snap install hugo --channel=extended
```

Quick Start

This section gives you a basic overview of how Hugo works and how you can create your first Hugo website. By proceeding, this section assumes that you have a working installation of Hugo on your machine.

1. To create a website called `mysite`, run:

   ```
   hugo new site mysite
   ```

   The above will create a folder called mysite which contains all Hugo essential files. Hugo comes with a hot-reload server that you can use in development.

2. Start the server.

   Hugo provides a hot-reload server for development purpose. All changes made to files will be reflected.

   ```
   hugo server -D
   ```

   Your Hugo website is now listening on https://localhost:1313

3. Add a theme.

   Here you go, your first Hugo website. You should notice however that your website is blank, to give life to your website, you must add a Hugo theme. Hundreds of Hugo themes are available online, Hugo official website has a repository of beautiful themes that can be found here.

   To add a theme, you need to save the theme under the *themes* folder. We will be using a certain theme called Ananke.

   ```
        cd mysite
        git init
        git submodule add https://github.com/theNewDynamic/gohugo-theme-
   ananke.git themes/ananke
   ```

4. Add theme to config.toml.

   After downloading the theme, you must tell Hugo the name of your theme. Edit the config.toml file and add the following:

   ```
   theme = "ananke"
   ```

   Hugo now recognizes your theme.

5. Add content.

   Hugo provides a command through which you can create new posts for your website. The following illustrates how to create a post titled "My first post" using Hugo command.

   ```
   hugo new posts/my-first-post.md
   ```

   The above command will generate a new file called my-first-post.md under the contents/post folder. You should notice that Hugo uses a markdown template for creating new posts. If you edit the newly created file, you should have something like this:

   ```
   ---
   title: "My First Post"
   date: 2019-03-26T08:47:11+01:00
   draft: true
   ---
   ```

Your content should go below the last hyphens. You must set the draft mode to "true" if you want the article to be published during deployment.

Alternatively, you can do this manually by creating a folder of your category under the contents folder. Inside this new folder you should create a .md file structured like above. Another method is to create another folder inside your category folder and create an index.md file within it. The following structures are valid ways of creating an Hugo post.

```
contents/posts/my-first-post.md
contents/posts/my-first-post/index.md
contents/about/index.md
contents/about.md
```

6. Preview website.

   Now that you have a theme and a post on your website, start the server and preview your website.

```
hugo server -D

     | EN
+------------------+----+
  Pages            | 10
  Paginator pages  |  0
  Non-page files   |  0
  Static files     |  3
  Processed images |  0
  Aliases          |  1
  Sitemaps         |  1
  Cleaned          |  0

Total in 11 ms
Watching for changes in /Users/bep/mysite/
{content,data,layouts,static,themes}
Watching for config changes in /Users/bep/mysite/config.toml
Environment: "development"
Serving pages from memory
Running in Fast Render Mode. For full rebuilds on change: hugo
 server --disableFastRender
Web Server is available at http://localhost:1313/ (bind address
 127.0.0.1)
Press Ctrl+C to stop
```

   Your website should now display some content.

For further configuration, read the chapter on structure.

Structure

The structure of a Hugo website is simple to understand. This sections explains each folders and how they can be configured to suit your complex needs.

```
.
### archetypes
### config.toml
### content
```

```
### data
### layouts
### static
### themes
```

archetypes

All types of contents used by your Hugo websites are defined. They are basically templates on how posts should look like.

config.toml

You should edit this file before deployment as it contains vital information about your website. A typical config file looks like:

```
baseURL = "https://example.org/"
languageCode = "en-us"
title = "My New Hugo Site"
theme = "ananke"
```

content

All contents for your website belong here. The sections and categories of your website determines how this folder can be structured.

data

Configuration files used by Hugo when generating your website are stored here. You can also create data templates here to generate data from dynamic contents.

layouts

HTML files that describe the appearance of your website are stored here. You should only edit these files if you know what you are doing.

static

All CSS, JavaScript, image, and other static files used by Hugo are stored here.

themes

All downloaded Hugo themes should be stored under this folder.

Related information

Hugo structure

Themes

Hugo themes, both free and paid, are available for use. Hugo official website host themes that covers blogs, portfolio, landing pages, and other categories. You can also find Hugo themes hosted by individuals or organizations under different kind of license.

Themes Installation

To install themes using Hugo, you simply have to download the theme and store it under the themes folder. If you have Git installed, you can clone the repository of the theme:

```
git clone https://github.com/theNewDynamic/gohugo-theme-ananke.git themes/
ananke
```

Make Hugo recognize the theme by adding the following to the config.toml file:

```
theme = ‘ananke’
```

Replace ‘ananke’ with theme name.

Example Site

Hugo theme developers follow a convention of shipping their themes with example sites which gives users a head start on how the contents of the site would look like. You should advantage of these demo sites by copying all the contents to the root directory of your website.

```
cp –rf themes/ananke/exampleSite/* .
```

Start the server after the above step, your website now displays demo contents which can be edited or deleted. This approach is advisable especially for beginners.

Theme Structure

Hugo theme have similar or exact structure to the default Hugo site directory. Your Hugo website without the content directory and few additional files like theme.toml, README, and LICENCE can serve as a theme for another Hugo website.

Related information

More themes

Deployment

Avoid using Hugo server in production, it is meant for development and testing purpose only. Hugo websites should be deployed like every other static websites. You can plug them into your choice of servers or you can deploy them without servers.

The Hugo command itself is symbolic, it does just what Hugo truly stands for.

Run the below command to generate your static website that works without a server:

```
hugo
```

Make sure you run the above command in the root folder of your Hugo website. After you run the command, a folder called public will be generated.

```
ls folder

about/
categories/
css/
font-awesome-4.7.0/
fonts/
img/
index.html
```

```
index.xml
js/
pricing/
sitemap.xml
tags/
```

This folder contains all the files needed for your website. You can now deploy this folder to cloud services like Netlify, GitHub Pages, etc. You can also plug them behind some server if you want to exercise some degree of control on the traffic of your site.

CI/CD Workflow

A better approach with Hugo is to put in place some form of CI/CD workflow. For GitHub Pages users, this is available for free and easy to implement. Contact your host to see if there is a way to deploy your website using CI/CD workflow.

Configuration

The config.yaml, config.toml, and config.json are files that Hugo searches for configuration. Even better, you can break down the configurations into subfolders under a parent folder called config/. Your config/ folder can look like below:

```
### config
#   ### _default
#   #   ### config.toml
#   #   ### languages.toml
#   #   ### menus.en.toml
#   #   ### menus.zh.toml
#   #   ### params.toml
#   ### production
#   #   ### config.toml
#   #   ### params.toml
#   ### staging
#       ### config.toml
#       ### params.toml
```

Note: The command *hugo server* uses _default/ for configuration and the command *hugo* uses production/ for configuration. To use staging/, run *hugo --environment staging* instead.

You can also specify custom configuration file by using the --config flag:

```
hugo --config debugconfig.toml
hugo --config a.toml,b.toml,c.toml
```

If your website contains multiple config files in different formats, Hugo gives them priority in the following order:

1. ./config.toml
2. ./config.yaml
3. ./config.json

Environment configuration is also available with Hugo. You can configure your website via environment variables which is useful for multiple deployments.

Use HUGO_* for config options and HUGO_PARAMS_* for config params. All environmental variables used as keys must be in upper case.

```
$ env HUGO_TITLE="My Title" hugo
```

Common Configuration Options

Hugo offers a rich set of configuration options that can make development easy. Below are some of these configurations that you can configure:

baseURL

This refers to your website url.

contentDir

This defaults to content.

copyright

Copyright notice typically displayed in the footer.

enableRobotsTXT

Configure the generation of `robots.txt` file here.

paginate

The number of posts to display per page. It defaults to `10`.

publishDir

Defaults to `public/`. This is the folder your static website will be written to if you run `hugo`.

theme

Selects theme to use.

title

Title for website.

watch

Monitor the filesystem for changes. Defaults to `false`.

Related information

Configuration options